Group of young creators

Chat Server - Client

تحت إشراف الإستاذ

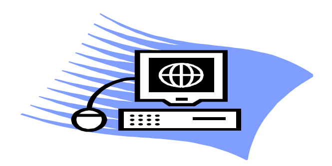
هارون عز الدين الربعي

إعداد طلاب نظم معلومات { مستوى ثاني }...

مالك هادي الهمامي ..

أحمد الدمشقي ..

عبد الباري الجبوري ..



. . .المقدمة . . .



بسم الله الرحمن الرحيمالحمدُ للهِ الذي خلق فعلَّم وجَعَلَ بعد العسر يسرًا و جعل الصبر مفتاح الفرج والنجاح ثمرة الجد والعمل شعور متدفق وفرحة عارمة وبسمة مرسومة على كل الشفاه ودموع حارة ومشاعر تختلج الفؤاد وشعور بالنجاح والتميز ،، ثم الصلاة والسلام والتمام للأشم ، رسولنا محمد نبينا المسدد ، ومن حذا لهديه أو التزم .....

. . أما بعد . .

نتقدم نحن مجموعة { Group of young creators } بتقديم هذا العمل المتواضع إلى أستاذنا و أخونا المحترم الأستاذ : (**هارون عز الدين الدبعي**) ..

و الشكر الخاص إلى كل من ساهم بإعانتا على إتمام هذا المشروع في لغة **vb.net** ....

- **محمد عبد السلام الحبيشي** .
- **جوانا عبدالرحمن الغامدي** .

ونتمنى من الله سبحانه وتعالى أن يوفقنا ويوفق زملائنا و زميلاتنا للنجاح الدائم والتفوق حتى نتخرج ويكون لنا الفخر في خدمة هذا الوطن وخدمة أهله لما يحبه الله ويرضاه ......





نبذة عن المشروع

Chat Server \ Client

وهو عبارة عن برنامج مصمم بإحدى لغات البرمجة وهي لغة **vb.net** ، ويستخدم هذا البرنامج للقيام بعملية إتصال بين أثنين مستخدمين لهذا البرنامج والمستخدم الأول قد يكون هو المسئول أو العميل والعكس وقبل البدء بعملية الإتصال للقيام بالمحادثة لابد أولاً من أيصال (كابل شبكة) بين كمبيوتر وكمبيوتر آخر وتغير رقم الآيبي لكمبيوتر العميل بنفس رقم الآيبي لكمبيوتر المسئول ،،، ويحتوي على أربع واجهات ..

الواجهة الأولى : وهي الواجهة الأساسية للمسئول Server .

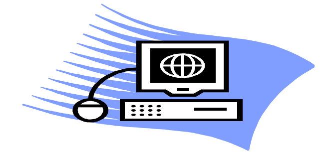
الواجهة الثانية : وهي واجهة التأمين ، ويتم الانتقال إليها من خلال الواجهة الأساسية للمسئولServer .

الواجهة الثالثة : وهي الواجهة الأساسية للعميل Client .

الواجهة الرابعة : وهي نفس واجهة التأمين لدى المسئول Server .

مميزات المشروع

يستطيع المستخدم لهذا البرنامج أثناء عملية المحادثة بينه وبين المستخدم الآخر في حالة انشغاله بأن يقوم بتأمين البرنامج برقم سري ويبقى البرنامج قيد التشغيل إلى أن يدخل المستخدم الرقم السري و يواصل المحادثة مع المستخدم الآخر ....

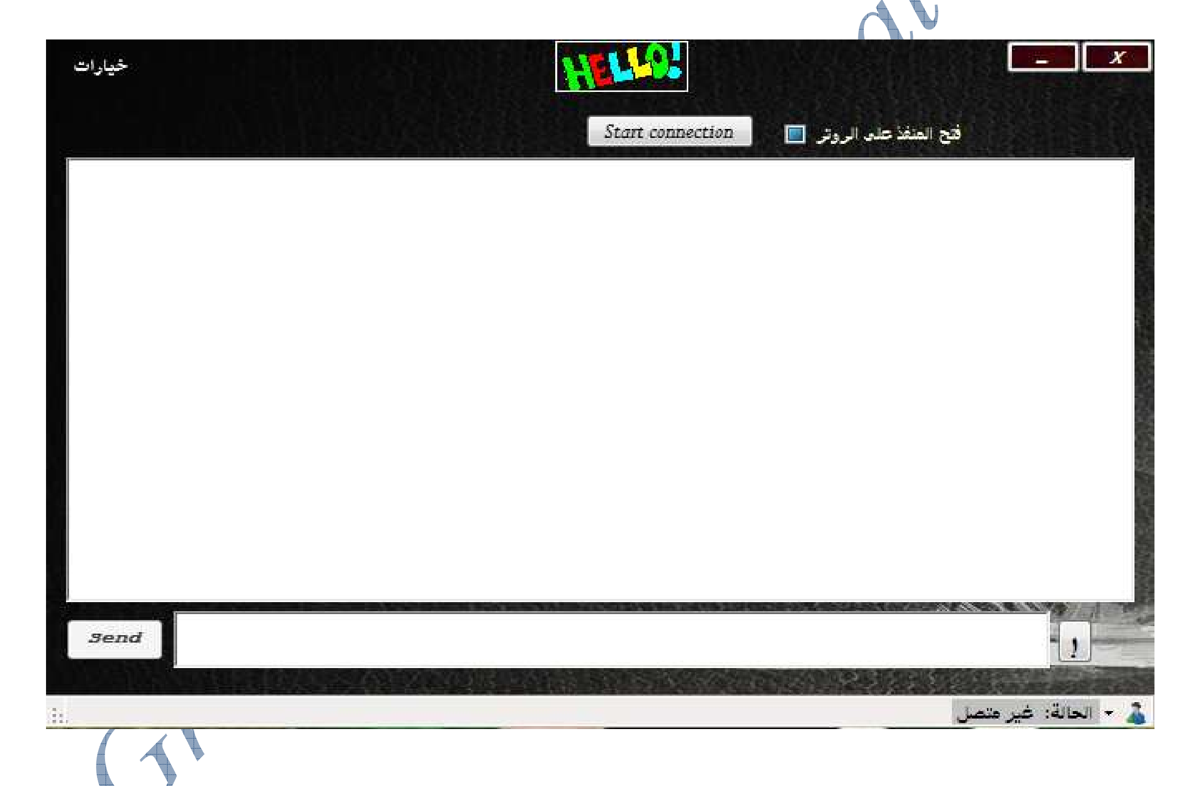


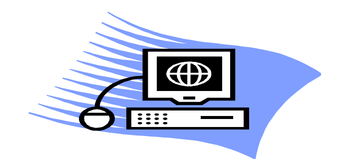


الشرح العام للمشروع

Server

الواجهة الأساسية للمسئول







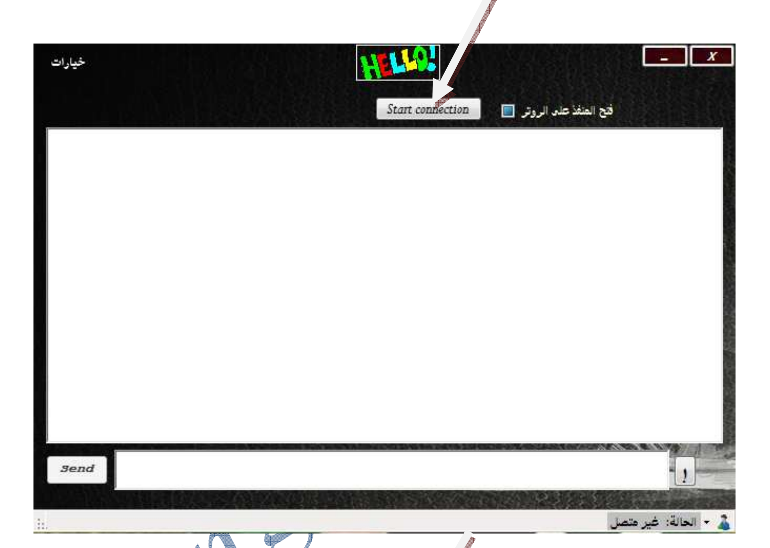
شرح الأدوات

تحتوي الواجهة الأساسية للمسئول على مجموعة من الأدوات ،، و سنقوم بشرح هاذي الأدوات بالتفصيل .....

فتح المنفذ على الروتر : وهو عبارة عن CheckBox1 الموجودة في قائمة (Toolbox) وبعد الضغط علية يتم الضغط على بوتون Start connection .

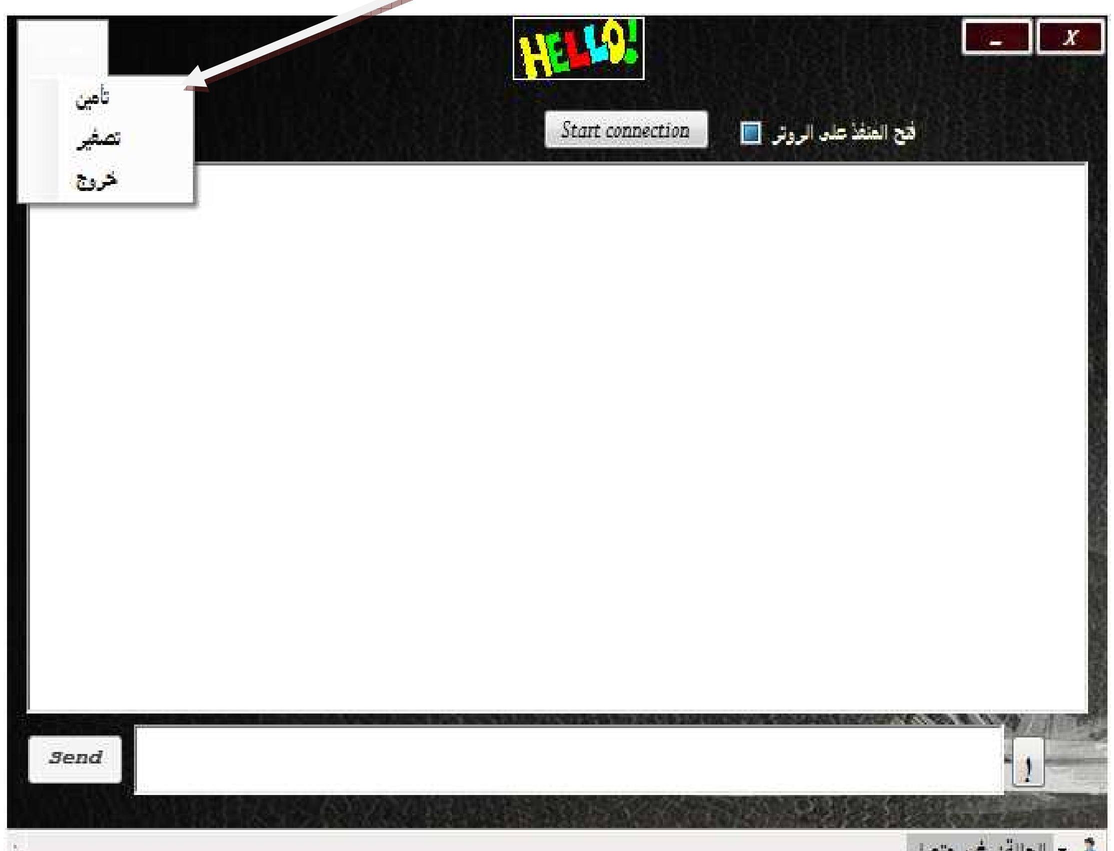


Start connection : وهو عبارة عن Button الموجودة في قائمة (Toolbox) عند الضغط علية تتم عملية الاتصال عبر رقم البورت ، ويظهر للمستخدم انه قد تم تشغيل المسئول بنجاح على العنوان (عنوان الآيبي للجهاز) في txtRecv .





MenuStrip1 : وهي عبارة عن أداة الموجودة في قائمة (Toolbox) تحتوي مجموعة من الخيارات التي يكتبها المصمم للبرنامج ويوجد لدينا في هذا الفورم مجموعة من الخيارات وهي (تأمين – أخفاء – خروج) .



txtRecv : وهو عبارة عن TextBox الموجود في قائمة (Toolbox) التي يظهر فيه الرسائل المرسلة من الـ Server والواردة من الـ Client .

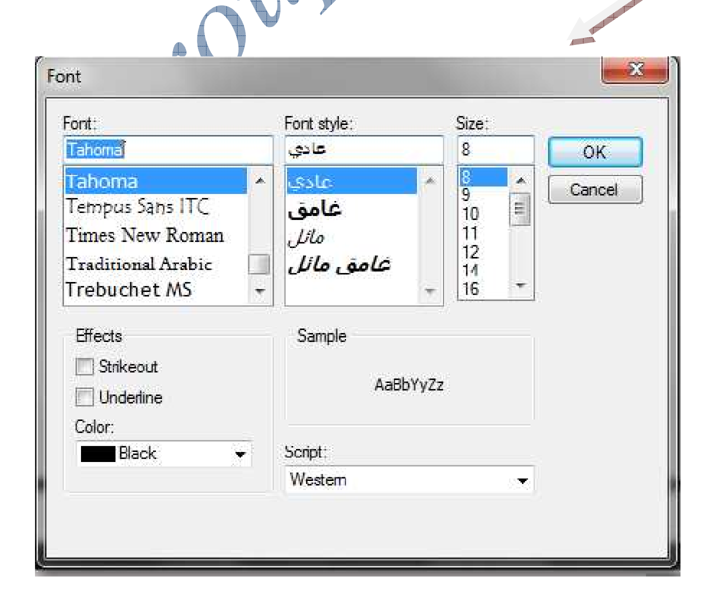




txtSend : وهو عبارة عن (TextBox) الموجودة في قائمة (Toolbox) التي يتم كتابة الرسائل فيه .



! : وهو عبارة عن Button الموجود في قائمة (Toolbox) لتغيير شكل ولون الخط في (txtRecv).





StatusStrip1 : وهو عبارة عن أداة الموجودة في قائمة (Toolbox) ، وكما موضح الشريط الذي يوجد أسفل الواجهة الرئيسية لل **Server** التي يظهر فيه الإحداث التي يقوم بها العميل في ال txtSend أثناء الكتابة .

Server

واجهة التأمين

ويتم الانتقال إلى شاشة التأمين عند الضغط على الخيار تأمين الموجود في قائمة الخيارات ، ولا يستطيع أي شخص أخر الدخول إلى الواجهة الرئيسة لـ **Server** إلا إذا قام بإدخال رقم سري صحيح كما أن البرنامج يبقى قيد التشغيل في حالة الانتقال إلى واجهة التأمين .





Password : وهو عبارة عن (Label) الموجود في قائمة (Toolbox) يستطيع المصمم أن يكتب بداخلة أي نص يريده .



TextBox : وهو التكست الموجود في قائمة (Toolbox) الذي يقوم الـ **Server**

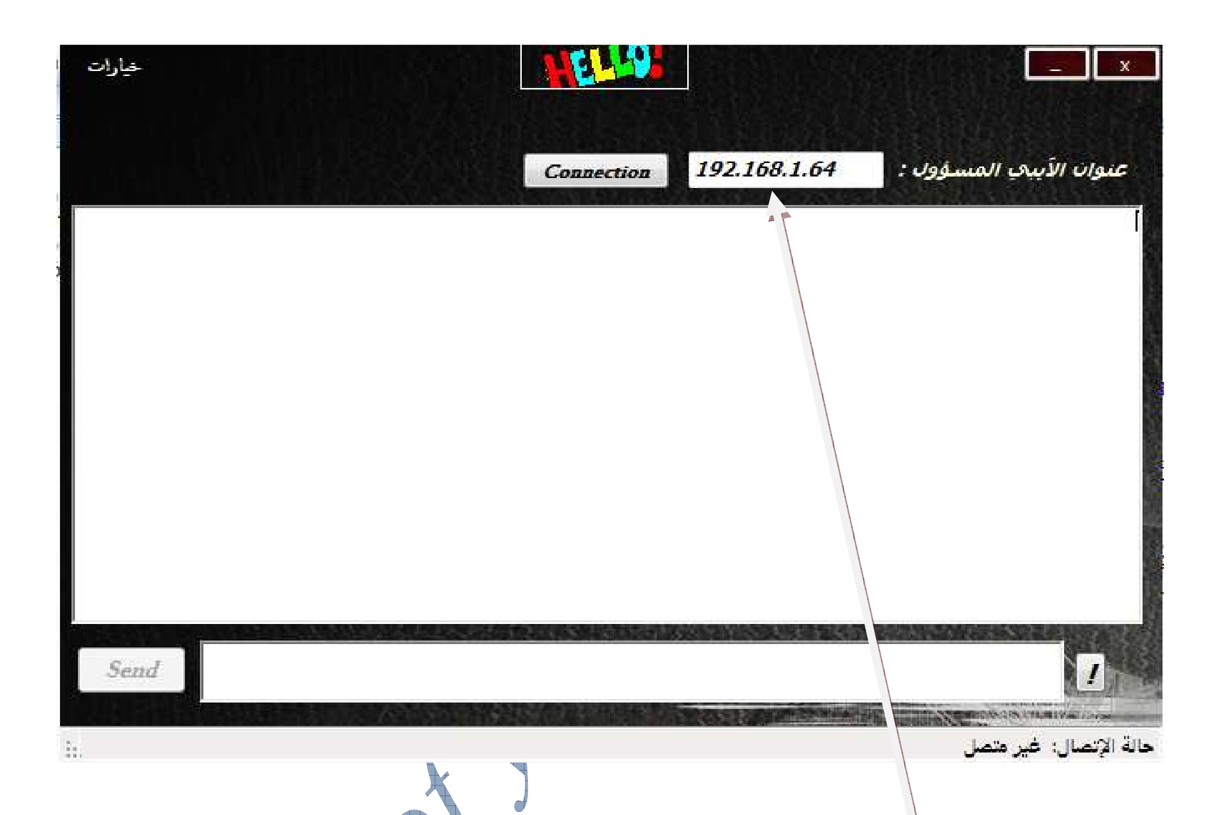
بكتابة الرقم السري بداخله لكي يتم رجوعه إلى الشاشة الرئيسية للـ **Server** .



Ok:وهو عبارة عن Button الموجود في قائمة (Toolbox) يتم الضغط علية بعد كتابة الرقم السري داخل الـ TextBox ، وفي حالة كتابة رقم سري غير صحيح تظهر رسالة (عذراً لا يمكنك الدخول) ،وفي حالة إدخال حروف تظهر رسالة (أدخل البيانات بشكل صحيح) ، وفي حالة إدخال رقم سري صحيح تظهر رسالة (مرحباً بك مرة أخرى) .

Client

الواجهة الأساسية للعميل



تحتوي الواجهة الأساسية للعميل على تكست عنوان الآيبي التي يتم كتابة الآيبي بداخل مع مراعاة أن يكون رقم الآيبي التي يكتب بداخلة نفس رقم الآيبي للمسئول ..





شرح الأكواد

أولاً : شرح أكواد الواجهة الأساسية للمسئول **Server** .

الإعلان عن المكتبات .

```
Imports System.Text
```

الشرح

System.Text : وتقوم هذه المكتبة بفك التشفير داخل الشبكة والتعامل مع البيانات النصية (String) ..

الإعلان عن المتغيرات (الإحداث) .

```
Public Class Form1

    Dim WithEvents MySock As BazSocket
    Dim WithEvents AcceptSock As BazSocket
    Private ImWriting As Boolean = False
    Private ImSupp As Boolean = False
```

الشرح

Dim WithEvents ⟶ تعريف حدث

MySock As BazSocket ⟶ يقوم بعملية الإتصال عبر المقبس (الإستماع للإتصال)

AcceptSock As BazSocket ⟶ يقوم بعملية الإتصال عبر المقبس (إرسال الإتصال)

ImWriting As Boolean = False ⟶ خاص بكتابة البيانات

ImSupp As Boolean = False ⟶ خاص بكتابة البيانات

القيام بعملية تهيئة للإتصال .

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load

        MySock = New BazSocket(Me)
        MySock.AlwaysRaiseClose = True

        End Sub
```



```
MySock = New BazSocket(Me)
MySock.AlwaysRaiseClose = True
```

هؤلاء السطرين يقومان بعملية تهيئة للإتصال والقيام بإتصال جديد ، بعد الإنتهاء من الإتصال السابق ...

الكود الخاص بزر الإرسال (**Send**) .

```
If AcceptSock.State = BazSocketState.Connected Then
     AcceptSock.Send(txtSend.Text)
     AddChatMessage(txtSend.Text, True)
     txtSend.Text = ""
     ImWriting = False
     ImSupp = False
     txtSend.Focus()
      End If
      End Sub
```

الكود الخاص بالـ StatusStrip1الشريط الذي يوجد أسفل واجهة المسؤول .

```
Private Sub ServerDataArrival(ByVal sender As Object, ByVal data() As Byte)

        Select Case Encoding.Unicode.GetString(data)
            Case "SWRITE"
            lbStatusConn.Text = "العميل يقوم بالكتابة "
            Case "SSUPPR"
            lbStatusConn.Text = " العميل يمسح الكتابة "
            Case "FINSUPP"
            lbStatusConn.Text = " متصل حاليا "
            Case Else
            lbStatusConn.Text = " متصل حاليا "
          AddChatMessage(Encoding.Unicode.GetString(data), False)

        End Select


        End Sub
```

الشرح

هذا الكود هو الكود الخاص بالشريط الذي يوجد أسفل فورم الواجهة الأساسية للمسئول Server والشريط هذا عبارة عن أداه تسمي **StatusStrip** الموجودة ضمن أدوات Toolbox واستخدمناها هنا في هذا الفورم لتبين للمسئول Server الإحداث الذي يقوم بها العميل Client في الواجهة الأساسية له داخل الـ txtSend أثناء عملية الكتابة ...

الكود الخاص بإستقبال الإتصال من العميل **Client** .

```
Private Sub MySock_Accepted(ByVal sender As System.Object, ByVal Request As AcceptRequest) Handles MySock.Accepted

        AcceptSock = New BazSocket(Me, Request)
        AcceptSock.AlwaysRaiseClose = True
        AddHandler AcceptSock.Closed, AddressOf Client_Close
        AddHandler AcceptSock.DataArrival, AddressOf ServerDataArrival
        txtRecv.AppendText(" تم اتصال العميل بنجاح  ==>>  " & AcceptSock.RemoteEP.Address.ToString() & vbCrLf)
        lbStatusConn.Text = "متصل الآن "

    End Sub
```

الشرح

هذا الكود يمكن المسئول Server من الإتصال بالعميل Client وفي حالة انه تم إتصال العميل بنجاح على نفس عنوان الآيبي للمسئول تظهر في شاشة الـtxtRecv الخاصة بالواجهة الأساسية للمسئول أنه (نفس رقم الآيبي للمسئول << == تم إتصال العميل بنجاح) ..

القيام بالإتصال (إتصال جديد) .

```
AcceptSock = New BazSocket(Me, Request)
```

بداية الإتصال، ويشترط للقيام بالإتصال أن يكون الإتصال مغلقاً في البداية .

```
AcceptSock.AlwaysRaiseClose = True
```

يشترط أن الذي يقوم ببداية عملية الإتصال هو المسؤول وأن يكون العميل غير متصل أي أن اتصاله مغلق .

```
AddHandler AcceptSock.Closed, AddressOf Client_Close
```

لكي يظهر في StatusStrip الشريط الذي أسفل الواجهة الأساسية للمسئول أنه متصل الآن في حالة إذا تم إتصاله بنجاح على عنوان الآيبي .

```
lbStatusConn.Text = "متصل الآن "
```

الرسائل أرسال وأستقبال الرسائل بين المسؤول والعميل .

```
Public Sub AddChatMessage(ByVal msg As String, ByVal meTalking As Boolean)
        If meTalking Then

         msg = ":المسؤول " + msg

        Else

        msg = ":العميل" + msg

         End If

        If msg = ":العميل" + "س1" Then

         msg = ":العميل " + "..[[ السلام عليكم ورحمة الله وبركاته ]].."

         End If

        If txtSend.Text = "س1" Then

         msg = ":المسؤول " + "..[[ السلام عليكم ورحمة الله وبركاته ]].."

         End If

        If txtRecv.Text.Length > txtRecv.MaxLength * 0.95 Then
          txtRecv.Text = txtRecv.Text.Remove(0,
        CInt(txtRecv.MaxLength / 2))

        End If

           txtRecv.Text += msg + ControlChars.Cr + ControlChars.Lf
           txtRecv.SelectionStart = txtRecv.Text.Length
           txtRecv.ScrollToCaret()


         End Sub
```



هذا الكود خاص بإرسال و إستقبال الرسائل بين المسئول والعميل ، وإضافة الاختصارات مثل

(س١ = السلام عليكم ورحمة الله وبركاته ، س٢ = وعليكم السلام ورحمة الله وبركاته) وأيضاً إعتبار أن البيانات التي تكتب بداخل التكست الخاص بالإرسال على أنها رسالة تظهر عند المسؤول والعميل.

الكود الخاص ببوتون الإتصال **Start connection** .

```
Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button2.Click

If MySock.State = BazSocketState.Disconnected Then

MySock.Listen(1007)

 txtRecv.AppendText ( "تم تشغيل المسؤول بنجاح على العنوان ==>> " )

 lbStatusConn.Text = " في حالة استماع "

If CheckBox1.Checked Then

UPnP.NAT.Discover()

txtRecv.AppendText(UPnP.NAT.GetExternalIP.ToString & vbNewLine)

UPnP.NAT.ForwardPort(1007, System.Net.Sockets.ProtocolType.Tcp, "chat")

  Else
   txtRecv.AppendText(MySock.LocalEP.Address.ToString & vbNewLine)
    End If
    Button2.Enabled = False

End If


End Sub
```

الشرح

يتم الضغط على بوتون الإتصال الموجود في الواجهة الرئيسية للمسؤول بعد الضغط على ال CheckBox (فتح المنفذ على الروتر) عن طريق الإستماع لمنفذ الروتر (1007) وبعد إتصال المسؤول بنجاح يظهر في الـtxtRecv الموجود في الواجهة الرئيسية للمسؤول أنه (==>>تم تشغيل المسؤول بنجاح على العنوان) .

```
If MySock.State = BazSocketState.Disconnected Then
MySock.Listen(1007)
txtRecv.AppendText(" تم تشغيل المسؤول بنجاح على العنوان ==>>")
```

وفي حالة تم إتصال المسؤول بنجاح يظهر في الشريط الموجود أسفل الواجهة الرئيسية للمسؤول (في حالة اتصال).

```
 lbStatusConn.Text =" في حالة استماع"
```

وبعد الضغط على الـ CheckBox (فتح المنفذ على الروتر) يتم الضغط على بوتون الإتصال .

```
If CheckBox1.Checked Then
UPnP.NAT.Discover()
txtRecv.AppendText(UPnP.NAT.GetExternalIP.ToString &
vbNewLine)
UPnP.NAT.ForwardPort(1007,
System.Net.Sockets.ProtocolType.Tcp, "chat")
  Else
txtRecv.AppendText(MySock.LocalEP.Address.ToString &
vbNewLine)
  End If
    Button2.Enabled = False
End If
End Sub
```

يستخدم هذا الكود لمعرفة أن العميل قد قام بقطع إتصاله وتظهر رسالة عند المسئول أن (العميل غير متصل الآن) ويظهر في الشريط الموجود أسفل شاشة العميل أنه (غير متصل) ...

```
Private Sub Client_Close(ByVal sender As Object, ByVal e As System.EventArgs)

        Dim sock As BazSocket = CType(sender, BazSocket)

        sock.Close()

        txtRecv.AppendText(" العميل غير متصل الآن" & vbNewLine)

        lbStatusConn.Text = " غير متصل "

         End Sub
```

يستخدم هذا الكود في تكست الإرسال لارسال ماتم كتابته في تكست الإرسال ..

```
Private Sub txtSend_KeyDown(ByVal sender As Object, ByVal e As
System.Windows.Forms.KeyEventArgs) HandlestxtSend.KeyDown
        If e.KeyCode = 8 AndAlso Not ImSupp Then

        AcceptSock.Send("SSUPPR")

        ImSupp = True

         End If

        End Sub
```

من خلال هذا الكود لا يستطيع كلاً من المسؤول والعميل الضغط على بوتون الإرسال الا إذا تمت الكتابة بداخل تكست الخاص بكتابة الرسائل ، ولا يمكن الكتابة بداخل تكست الإرسال الا إذا تمت عملية الإتصال بين كلاً من العميل والمسئول ..

```
Private Sub txtSend_TextChanged(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles txtSend.TextChanged

Button1.Enabled = txtSend.TextLength > 0

If txtSend.TextLength > 0 Then

If Not ImWriting AndAlso AcceptSock.State = BazSocketState.Connected  Then

AcceptSock.Send("SWRITE")

ImWriting = True

 End If

ElseIf txtSend.TextLength = 0 Then

If ImSupp AndAlso AcceptSock.State = BazSocketState.Connected Then

 AcceptSock.Send("FINSUPP")

ImWriting = False

ImSupp = False

txtSend.Focus()

  End If

 End If
```

```
Private Sub Button3_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button3.Click

Me.Close()

End Sub
```

الكود الخاص بخيار الخروج من البرنامج أو البوتون X

```
Private Sub Button4_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button4.Click

  Me.WindowState = 1

End Sub
```

الكود الخاص بخيار تصغير الواجهة الأساسية للـ **Server** و الـ **Client** أو البوتون _

الكود الخاص بخيار تأمين ضمن قائمة الخيارات الموجود في الواجهة الأساسية للـ **Server** و الـ **Client** للانتقال إلى واجهة التأمين .

```
Private Sub تأمينToolStripMenuItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
تأمينToolStripMenuItem.Click

  Me.Hide()

 Form2.Show()

 End Sub
```

الكود الخاص ببوتون ! الموجود في الواجهة الأساسية للـ **Server** و الـ **Client** ، للإنتقال إلى شاشة إعداد تغيير شكل ولون الخط بداخل الـ txtRecv ..

```
Private Sub Button6_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button6.Click

 FontDialog1.ShowColor = True

 FontDialog1.ShowDialog()

With txtRecv

.Font = FontDialog1.Font

.ForeColor = FontDialog1.Color

 End With


 End Sub


 End Sub
```

الكود الخاص بواجهة التأمين للمسئول**Server** و للعميل **Client** .

```
Public Class Form2

Private Sub textbox1_KeyPress(ByVal sender As Object, ByVal e As
System.Windows.Forms.KeyPressEventArgs) Handles TextBox1.KeyPress

If Asc(e.KeyChar) = 13 Then

        Try

Dim a As Integer
a = TextBox1.Text

If a = "111" Then
    MsgBox(" مرحباً بك مرة أخرى")
  TextBox1.Clear()
     Me.Hide()
     Form1.Show()
     Else
 MsgBox("عذراً لا يمكنك الدخول", MsgBoxStyle.Critical, "Group of young creators")
                    TextBox1.Clear()
                End If
            Catch ex As Exception
                MsgBox(" أدخل البيانات بشكل صحيح")

                TextBox1.Clear()
            End Try

        End If
    End Sub
```


الشرح

عندما يدخل المستخدم الرقم السري في خانة الباسورد التكست بوكس التابع له بعد ذلك يُمكن المستخدم أن يضغط مره واحد على زر **Enter** الموجود في لوحة المفاتيح في الكمبيوتر .

```
If Asc(e.KeyChar) = 13 Then
```

الكود الخاص ببوتون **Ok** ...

```
Private Sub Button1_Click_1(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click

      Try

Dim a As Integer
a = TextBox1.Text

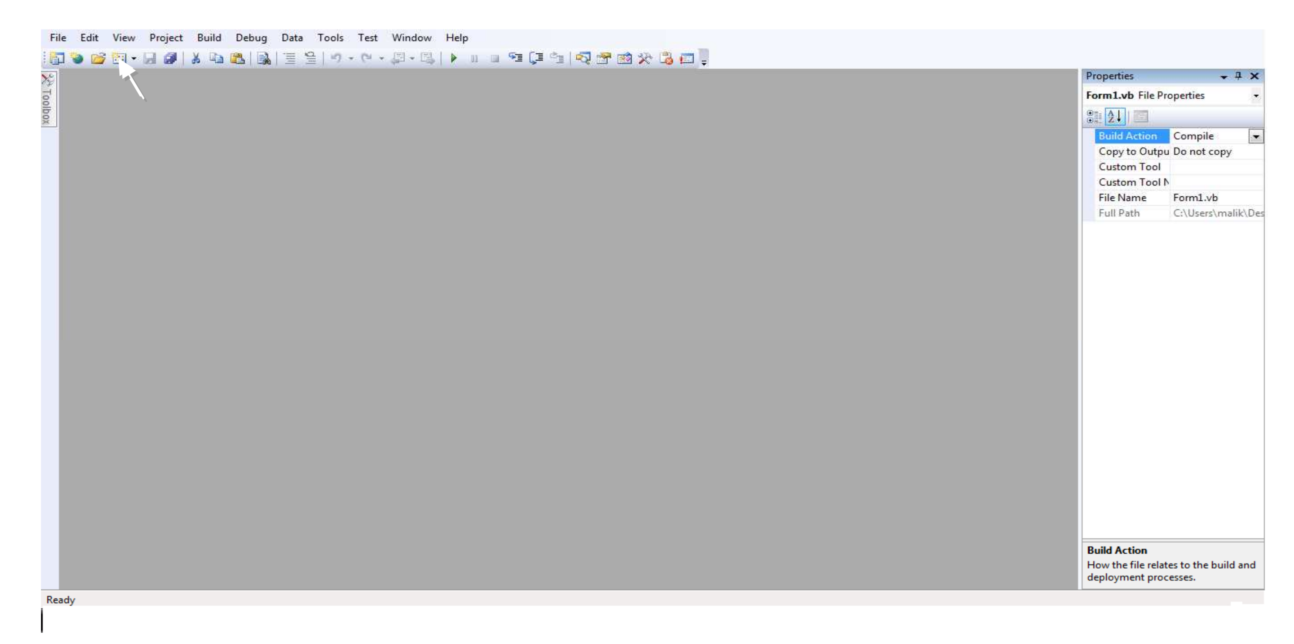
If a = "111" Then
MsgBox(" مرحباً بك مره أخرى")
TextBox1.Clear()
  Me.Hide()
  Form1.Show()
 Else
MsgBox("عذراً لا يمكنك الدخول", MsgBoxStyle.Critical, "Group of young creators")
                  TextBox1.Clear()
              End If
          Catch ex As Exception
              MsgBox(" أدخل البيانات بشكل صحيح")

              TextBox1.Clear()
          End Try
      End Sub
  End Class
```

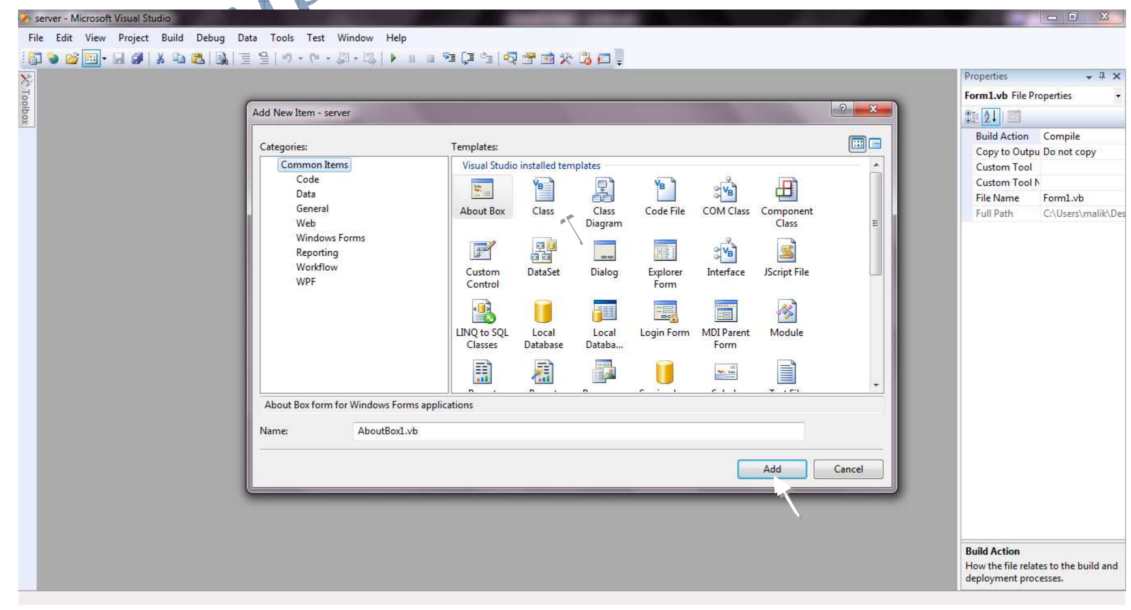
أضافة AcceptRequest.vb للمشروع ...

الــ AcceptRequest.vb :- هي عبارة عن Class.vb (كلاس) ، يتم أستدعائها كما هو موضح في الصور التالية ، وقد استخدمنها في المسئول Server و العميل Client .

(صوره ١)



(صوره ٢)



ويحتوي الـ **AcceptRequest.vb** على الكود التالي ....

```
Imports System.Net.Sockets
Imports System.Net
Public Class AcceptRequest
 Private m_Socket As Socket
 Private m_Stream As NetworkStream

Friend Sub New(ByVal sock As Socket, ByVal stream As NetworkStream)
        m_Socket = sock
        m_Stream = stream

End Sub

Friend ReadOnly Property Sock() As Socket
 Get
       Return m_Socket

End Get

 End Property

Friend ReadOnly Property Stream() As NetworkStream
    Get

Return m_Stream

 End Get

    End Property

    Public ReadOnly Property LocalEP() As IPEndPoint
        Get

   Return CType(m_Socket.LocalEndPoint, IPEndPoint)

  End Get

    End Property

    Public ReadOnly Property RemoteEP() As IPEndPoint
        Get

    Return CType(m_Socket.RemoteEndPoint, IPEndPoint)

    End Get

   End Property

End Class
```



AcceptRequest.vb : تحتوي على حالات الإتصال ، وحالة الإتصال مع الآيبي لجهازك ......

BazSocket.vb: وتحتوي على كود يعمل على برمجة أداة السوكت **Socket** داخل لغة **Vb.net** وهو عبارة عن **Class.vb** (كلاس) يتم استدعائه كما تم ايضاحة في الصور السابقه ...


```
Imports System.Net
Imports System.Net.Sockets
Imports System.ComponentModel
Imports System.Text
Public Class BazSocket

#Region " Membres "
    'TODO: Metre en property ?
    Private Const BACKLOG As Integer = 8

    Private m_RecvSize As Integer   'Taille du buffer pour recevoir
les données
    Private m_RecvBuffer() As Byte  'Buffer pour recevoir les données

    Private m_SendSize As Integer   'Taille du buffer pour envoyer
les données
    Private m_SendBuffer() As Byte  'Buffer pour envoyer les données

    Private m_Sync As ISynchronizeInvoke 'Merci à Xya ;)

    Private m_Socket As Socket
    Private m_Stream As NetworkStream
    Private m_State As BazSocketState  'Etat du socket

    Private m_LocalEP As IPEndPoint
    Private m_RemoteEP As IPEndPoint
    Private m_RemotePort As Integer

    Private m_Init As Boolean       'Est-ce que le socket est
initialisé ou pas ?
    'Voir Property AsyncEvent
    Private m_AsyncEvent As Boolean
    'Voir Property AlwaysRaiseClose
    Private m_AlwaysRaiseClose As Boolean

    Private AcceptSync As Object '}
```

```
    Private SendSync As Object   '} Object servant pour le SyncLock
des threads
    Private RecvSync As Object   '}
#End Region

#Region " Constructeur "
    'Constructeur par default
    Public Sub New()
        Me.New(Nothing)
    End Sub
    'Constructeur
    Public Sub New(ByRef Sync As ISynchronizeInvoke)
        m_Sync = Sync
        InitInvoke()


        'Valeur par default
        m_AlwaysRaiseClose = False
        m_AsyncEvent = True

        AcceptSync = New Object
        SendSync = New Object
        RecvSync = New Object

        CreateSocket()
        InitBuffers()

        OnStateChanged(BazSocketState.Disconnected)
    End Sub

    'Creation du socket pour accepter une nouvelle connection
    Public Sub New(ByRef Sync As ISynchronizeInvoke, ByRef Request As
AcceptRequest)
        If Request.Sock Is Nothing OrElse Request.Sock.Connected =
False OrElse Request.Stream Is Nothing Then
            m_Init = False
            Return
        End If

        m_Sync = Sync
        InitInvoke()

        m_AlwaysRaiseClose = False
        m_AsyncEvent = True

        m_Socket = Request.Sock
        m_Stream = Request.Stream
        m_State = BazSocketState.Connected
        InitBuffers()

        AcceptSync = New Object
        SendSync = New Object
        RecvSync = New Object

        m_Init = True

        OnStateChanged(BazSocketState.Connected)

        'On commence a recevoir les données qui pourraient arriver.
        m_Stream.BeginRead(m_RecvBuffer, 0, m_RecvSize, New
AsyncCallback(AddressOf Receive_CallBack), m_Socket)
```

```
    End Sub
#End Region

#Region " Events "
    Public Event StateChanged As StateChangedEventHandler
    Public Event Connected As EventHandler
    Public Event ConnectionFailed As ErrorEventHandler
    Public Event Listening As EventHandler
    Public Event ListenFailed As ErrorEventHandler
    Public Event Accepted As AcceptedEventHandler
    Public Event AcceptFailed As ErrorEventHandler
    Public Event DataArrival As DataArrivalEventHandler
    Public Event SendProgress As SendProgressEventHandler
    Public Event SendComplete As SendCompleteEventHandler
    Public Event Closed As EventHandler
    Public Event ThreadException As ErrorEventHandler
#End Region

#Region " -- Cross-Threading -- "
    'Merci à Xya pour son aide qui ma permit de régler le problème de
Cross-Threading !!

    'Délégués pour appeler les sub grace a m_Sync.Invoke
    Private Delegate Sub _StateChanged(ByVal state As BazSocketState)
    Private Delegate Sub _Connected()
    Private Delegate Sub _ConnectionFailed(ByVal ex As Exception)
    Private Delegate Sub _Listening()
    Private Delegate Sub _ListenFailed(ByVal ex As Exception)
    Private Delegate Sub _Accepted(ByVal Request As AcceptRequest)
    Private Delegate Sub _AcceptFailed(ByVal ex As Exception)
    Private Delegate Sub _DataArrival(ByVal data() As Byte)
    Private Delegate Sub _SendProgress(ByVal current As Integer,
ByVal total As Integer)
    Private Delegate Sub _SendComplete(ByVal total As Integer)
    Private Delegate Sub _Close()
    Private Delegate Sub _ThreadException(ByVal ex As Exception)

    'Les Délégués
    Private CallStateChanged As _StateChanged
    Private CallConnected As _Connected
    Private CallConnectionFailed As _ConnectionFailed
    Private CallListening As _Listening
    Private CallListenFailed As _ListenFailed
    Private CallAccepted As _Accepted
    Private CallAcceptFailed As _AcceptFailed
    Private CallDataArrival As _DataArrival
    Private CallSendProgress As _SendProgress
    Private CallSendComplete As _SendComplete
    Private CallClose As _Close
    Private CallThreadException As _ThreadException

    Private Sub InitInvoke()
        CallStateChanged = New _StateChanged(AddressOf
Sync_StateChanged)
        CallConnected = New _Connected(AddressOf Sync_Connected)
        CallConnectionFailed = New _ConnectionFailed(AddressOf
Sync_ConnectionFailed)
        CallListening = New _Listening(AddressOf Sync_Listening)
        CallListenFailed = New _ListenFailed(AddressOf
Sync_ListenFailed)
        CallAccepted = New _Accepted(AddressOf Sync_Accepted)
```

```
        CallAcceptFailed = New _AcceptFailed(AddressOf
Sync_AcceptFailed)
        CallDataArrival = New _DataArrival(AddressOf
Sync_DataArrival)
        CallSendProgress = New _SendProgress(AddressOf
Sync_SendProgress)
        CallSendComplete = New _SendComplete(AddressOf
Sync_SendComplete)
        CallClose = New _Close(AddressOf Sync_Closed)
        CallThreadException = New _ThreadException(AddressOf
Sync_ThreadException)
    End Sub

    'Sert a regarder si on a besoin d'apeller l'Event depuis un autre
thread.
    Private Function InvokeRequired() As Boolean
        If m_Sync IsNot Nothing AndAlso m_Sync.InvokeRequired Then
            Return True
        Else
            Return False
        End If
    End Function

    'Evenements appelé par les threads.
    Private Sub OnStateChanged(ByVal state As BazSocketState)
        Try
            If InvokeRequired() Then
                If m_AsyncEvent Then
          m_Sync.BeginInvoke(CallStateChanged, New Object() {state})
                Else
          m_Sync.Invoke(CallStateChanged, New Object() {state})
                End If
            Else
                Sync_StateChanged(state)
            End If
        Catch ex As Exception
            OnThreadException(ex)
        End Try
    End Sub
    Private Sub OnConnected()
        Try
            If InvokeRequired() Then
                If m_AsyncEvent Then
                    m_Sync.BeginInvoke(CallConnected, Nothing)
                Else
                    m_Sync.Invoke(CallConnected, Nothing)
                End If
            Else
                Sync_Connected()
            End If
        Catch ex As Exception
            OnThreadException(ex)
        End Try
    End Sub
    Private Sub OnConnectionFailed(ByVal ex As Exception)
        Try
            If InvokeRequired() Then
                If m_AsyncEvent Then
       m_Sync.BeginInvoke(CallConnectionFailed, New Object() {ex})
                Else
        m_Sync.Invoke(CallConnectionFailed, New Object() {ex})
```

```
                End If
            Else
                Sync_ConnectionFailed(ex)
            End If
        Catch exp As Exception
            OnThreadException(exp)
        End Try
    End Sub
    Private Sub OnListening()
        Try
            If InvokeRequired() Then
                If m_AsyncEvent Then
                    m_Sync.BeginInvoke(CallListening, Nothing)
                Else
                    m_Sync.Invoke(CallListening, Nothing)
                End If
            Else
                Sync_Listening()
            End If
        Catch ex As Exception
            OnThreadException(ex)
        End Try
    End Sub
    Private Sub OnListenFailed(ByVal ex As Exception)
        Try
            If InvokeRequired() Then
                If m_AsyncEvent Then
     m_Sync.BeginInvoke(CallListenFailed, New Object() {ex})
                Else
      m_Sync.Invoke(CallListenFailed, New Object() {ex})
                End If
            Else
                Sync_ListenFailed(ex)
            End If
        Catch exp As Exception
            OnThreadException(exp)
        End Try
    End Sub
    Private Sub OnAccepted(ByVal Request As AcceptRequest)
        Try
            If InvokeRequired() Then
                If m_AsyncEvent Then
      m_Sync.BeginInvoke(CallAccepted, New Object() {Request})
                Else
       m_Sync.Invoke(CallAccepted, New Object() {Request})
                End If
            Else
                Sync_Accepted(Request)
            End If
        Catch ex As Exception
            OnThreadException(ex)
        End Try
    End Sub
    Private Sub OnAcceptFailed(ByVal ex As Exception)
        Try
            If InvokeRequired() Then
                If m_AsyncEvent Then
       m_Sync.BeginInvoke(CallAcceptFailed, New Object() {ex})
                Else
   m_Sync.BeginInvoke(CallAcceptFailed, New Object() {ex})
                End If
```

```
                Else
                    Sync_AcceptFailed(ex)
                End If
            Catch exp As Exception
                OnThreadException(exp)
            End Try
        End Sub
        Private Sub OnDataArrival(ByVal data() As Byte)
            Try
                If InvokeRequired() Then
                    If m_AsyncEvent Then
              m_Sync.BeginInvoke(CallDataArrival, New Object() {data})
                    Else
             m_Sync.Invoke(CallDataArrival, New Object() {data})
                    End If
                Else
                    Sync_DataArrival(data)
                End If
            Catch ex As Exception
                OnThreadException(ex)
            End Try
        End Sub
        Private Sub OnSendProgress(ByVal current As Integer, ByVal total
    As Integer)
            Try
                If InvokeRequired() Then
                    If m_AsyncEvent Then
       m_Sync.BeginIvoke(CallSendProgress, New Object() {current, total})
                    Else
        m_Sync.Invoke(CallSendProgress, New Object() {current, total})
                    End If
                Else
                    Sync_SendProgress(current, total)
                End If
            Catch ex As Exception
                OnThreadException(ex)
            End Try
        End Sub
        Private Sub OnSendComplete(ByVal total As Integer)
            Try
                If InvokeRequired() Then
                    If m_AsyncEvent Then
    m_Sync.BeginInvoke(CallSendComplete, New Object() {total})
                    Else
      m_Sync.Invoke(CallSendComplete, New Object() {total})
                    End If
                Else
                    Sync_SendComplete(total)
                End If
            Catch ex As Exception
                OnThreadException(ex)
            End Try
        End Sub
        Private Sub OnClosed()
            Try
                If InvokeRequired() Then
                    If m_AsyncEvent Then
                        m_Sync.BeginInvoke(CallClose, Nothing)
                    Else
                        m_Sync.Invoke(CallClose, Nothing)
                    End If
```

```
                Else
                    Sync_Closed()
                End If
            Catch ex As Exception
                OnThreadException(ex)
            End Try
        End Sub
        Private Sub OnThreadException(ByVal ex As Exception)
            If InvokeRequired() Then
                If m_AsyncEvent Then
         m_Sync.BeginInvoke(CallThreadException, New Object() {ex})
                Else
          m_Sync.Invoke(CallThreadException, New Object() {ex})
                End If
            Else
                Sync_ThreadException(ex)
            End If
        End Sub

   'Fonction appeler dans le thread m_Sync qui vont declencher les
Events
        Private Sub Sync_StateChanged(ByVal state As BazSocketState)
            RaiseEvent StateChanged(Me, state)
        End Sub
        Private Sub Sync_Connected()
            RaiseEvent Connected(Me, EventArgs.Empty)
        End Sub
        Private Sub Sync_ConnectionFailed(ByVal ex As Exception)
            RaiseEvent ConnectionFailed(Me, ex)
        End Sub
        Private Sub Sync_Listening()
            RaiseEvent Listening(Me, EventArgs.Empty)
        End Sub
        Private Sub Sync_ListenFailed(ByVal ex As Exception)
            RaiseEvent ListenFailed(Me, ex)
        End Sub
        Private Sub Sync_Accepted(ByVal Request As AcceptRequest)
            RaiseEvent Accepted(Me, Request)
        End Sub
        Private Sub Sync_AcceptFailed(ByVal ex As Exception)
            RaiseEvent AcceptFailed(Me, ex)
        End Sub
        Private Sub Sync_DataArrival(ByVal data() As Byte)
            RaiseEvent DataArrival(Me, data)
        End Sub
        Private Sub Sync_SendProgress(ByVal current As Integer, ByVal
total As Integer)
            RaiseEvent SendProgress(Me, current, total)
        End Sub
        Private Sub Sync_SendComplete(ByVal total As Integer)
            RaiseEvent SendComplete(Me, total)
        End Sub
        Private Sub Sync_Closed()
            RaiseEvent Closed(Me, EventArgs.Empty)
        End Sub
        Private Sub Sync_ThreadException(ByVal ex As Exception)
            RaiseEvent ThreadException(Me, ex)
        End Sub
#End Region

#Region " Proprietés "
```

```
    Public ReadOnly Property Initialised() As Boolean
        Get
            Return m_Init
        End Get
    End Property

    'Etat du socket
    Public ReadOnly Property State() As BazSocketState
        Get
            Return m_State
        End Get
    End Property

    Public Property LocalEP() As IPEndPoint
        Get
            If m_State <> BazSocketState.Disconnected Then
                Return CType(m_Socket.LocalEndPoint, IPEndPoint)
            Else
                Return m_LocalEP
            End If
        End Get
        Set(ByVal Value As IPEndPoint)
            If m_State = BazSocketState.Disconnected Then
                m_LocalEP = Value
            End If
        End Set
    End Property
    Public Property RemoteEP() As IPEndPoint
        Get
            If m_State = BazSocketState.Connected OrElse m_State =
BazSocketState.Connecting Then
                Return CType(m_Socket.RemoteEndPoint, IPEndPoint)
            Else
                Return m_RemoteEP
            End If
        End Get
        Set(ByVal Value As IPEndPoint)
            If m_State = BazSocketState.Disconnected Then
                m_RemoteEP = Value
            End If
        End Set
    End Property

    'Si True,  L'evenement Close est déclenché quand vous appelez
Disconnect.
    'Si False, L'evenement Close est déclenché uniquement quand la
connection est fermé a distance. Comme Winsock de VB6 (Defaut)
    Public Property AlwaysRaiseClose() As Boolean
        Get
            Return m_AlwaysRaiseClose
        End Get
        Set(ByVal value As Boolean)
            m_AlwaysRaiseClose = value
        End Set
    End Property

    'Si True,  Les Evenements sont lancés chaque fois qu'il ce
produise. Comme Winsock de VB6 (Defaut)
    'Si False, Les Evenements sont mit en attente s'il y en a deja un
qui est entrin de s'executer. Ca permet de traiter les données dans
l'ordre ou elle arrive.
```

```
    Public Property AsyncEvent() As Boolean
        Get
            Return m_AsyncEvent
        End Get
        Set(ByVal value As Boolean)
            m_AsyncEvent = value
        End Set
    End Property

    'Permet de regler la taille des buffers d'envoie et de reception.
    Public Property SendBufferSize() As Integer
        Get
            Return m_SendSize
        End Get
        Set(ByVal value As Integer)
      If m_Init AndAlso m_State = BazSocketState.Disconnected Then
                m_SendSize = value
                m_Socket.SendBufferSize = m_SendSize
                ReDim m_SendBuffer(m_SendSize - 1)
            End If
        End Set
    End Property
    Public Property ReceiveBufferSize() As Integer
        Get
            Return m_RecvSize
        End Get
        Set(ByVal value As Integer)
    If m_Init AndAlso m_State = BazSocketState.Disconnected Then
                m_RecvSize = value
                m_Socket.ReceiveBufferSize = m_RecvSize
                ReDim m_RecvBuffer(m_RecvSize - 1)
            End If
        End Set
    End Property

    'Merci à Xya ;)
    'Pour le Cross-Threading
    Public Property SynchronizingObject() As ISynchronizeInvoke
        Get
            Return m_Sync
        End Get
        Set(ByVal value As ISynchronizeInvoke)
            m_Sync = value
        End Set
    End Property
#End Region

#Region " Public Methodes "
    Public Sub Connect(ByVal RemoteHost As String, ByVal RemotePort
As Integer)
        If m_State = BazSocketState.Disconnected AndAlso m_Socket
IsNot Nothing Then
            Try
                ' C'est mieu d'utilisé BeginGetHostEntry() pour ne
pas bloqué le thread en cours
                m_RemotePort = RemotePort 'TODO: ca serai plus beau
de passer ca dans l'objet state de GetHostEntry_CallBack
                m_State = BazSocketState.Resolving
                OnStateChanged(BazSocketState.Resolving)
                Dns.BeginGetHostEntry(RemoteHost, New
AsyncCallback(AddressOf GetHostEntry_CallBack), m_Socket)
```

```
            Catch ex As Exception
                m_State = BazSocketState.Disconnected
                OnStateChanged(BazSocketState.Disconnected)
                OnConnectionFailed(ex)
            End Try
        End If
    End Sub
    Public Sub Connect()
        Me.Connect(m_LocalEP, m_RemoteEP)

 End SubPublic Sub Connect(ByVal LocalEP As IPEndPoint, ByVal
RemoteEP As IPEndPoint)
        If m_State = BazSocketState.Disconnected AndAlso m_Init
AndAlso m_RemoteEP IsNot Nothing Then
            Try
                m_State = BazSocketState.Connecting
                OnStateChanged(BazSocketState.Connecting)

                If m_LocalEP IsNot Nothing Then
                    m_Socket.Bind(m_LocalEP)
                End If

                m_Socket.BeginConnect(m_RemoteEP, New
AsyncCallback(AddressOf Connect_CallBack), m_Socket)
            Catch ex As Exception
                m_State = BazSocketState.Disconnected
                OnStateChanged(BazSocketState.Disconnected)
                OnConnectionFailed(ex)
            End Try
        End If
    End Sub

    Public Sub Disconnect()
        If m_State <> BazSocketState.Disconnected Then

            If m_AlwaysRaiseClose Then
                DisconnectSocket(True)
            Else
                DisconnectSocket(False)
            End If
        End If
    End Sub

    Public Sub Close()
        If m_State <> BazSocketState.Disconnected AndAlso
m_AlwaysRaiseClose Then
            CloseSocket()
            OnClosed()
        Else
            CloseSocket()
        End If
    End Sub


    Public Function Listen() As Boolean
        Me.Listen(m_LocalEP)
    End Function
    Public Sub Listen(ByVal LocalIP As String, ByVal LocalPort As
Integer)
        Try
            m_LocalEP = DefaultLocalEP(LocalIP, LocalPort)
```

```
        Catch ex As Exception
            OnListenFailed(ex)
            Return
        End Try


        Me.Listen(m_LocalEP)
    End Sub
    Public Sub Listen(ByVal LocalPort As Integer)
        Try
            m_LocalEP = DefaultLocalEP(LocalPort)
        Catch ex As Exception
            OnListenFailed(ex)
            Return
        End Try

        Me.Listen(m_LocalEP)
    End Sub
    Public Sub Listen(ByVal LocalEP As IPEndPoint)
        If m_State = BazSocketState.Disconnected AndAlso m_Init Then
            Try
                m_Socket.Bind(LocalEP)

                m_Socket.Listen(BACKLOG)

m_Socket.BeginAccept(New AsyncCallback(AddressOf Accept_CallBack),
m_Socket)

                m_State = BazSocketState.Listening
                OnStateChanged(BazSocketState.Listening)
                OnListening()
            Catch ex As Exception
                m_State = BazSocketState.Disconnected
                OnStateChanged(BazSocketState.Disconnected)
                OnListenFailed(ex)
            End Try

        End If
    End Sub

    Public Sub Send(ByVal data() As Byte)
        If data Is Nothing Then Return
        If data.Length = 0 Then Return

        If m_State = BazSocketState.Connected Then
            m_Stream.BeginWrite(data, 0, data.Length, New
AsyncCallback(AddressOf Send_CallBack), m_Socket)
        End If
    End Sub

    'Prend en parametre une string pour facilité les choses.
    Public Sub Send(ByVal data As String)
        If m_State = BazSocketState.Connected Then
            Dim ByteData() As Byte = Encoding.Unicode.GetBytes(data)
            Me.Send(ByteData)
        End If
    End Sub
#End Region

#Region " CallBack Methode "
    Private Sub GetHostEntry_CallBack(ByVal Ar As IAsyncResult)
```

```
        Dim AsyncSocket As Socket = CType(Ar.AsyncState, Socket)

        'Regarde si le socket Existe encore, Sinon on quite
        If Not Object.ReferenceEquals(AsyncSocket, m_Socket) Then
            Return
        End If

        Try
            Dim HostEntry As IPHostEntry = Dns.EndGetHostEntry(Ar)
            'On boucle sur la liste des address retourné pour en
trouver une en IPV4 car la 1er de la liste est une IPV6 sous Vista ;)
            For Each Ip As IPAddress In HostEntry.AddressList
                If Ip.AddressFamily = AddressFamily.InterNetwork Then
                    m_RemoteEP = New IPEndPoint(Ip, m_RemotePort)
                    m_State = BazSocketState.Disconnected
                    Me.Connect(m_LocalEP, m_RemoteEP)
                    Return
                End If
            Next

            ' Si on arrive ici, c'est qu'il n'y a pas d'address IPV4
pour cet Host.
            m_State = BazSocketState.Disconnected
            OnStateChanged(BazSocketState.Disconnected)
            OnConnectionFailed(New
SocketException(SocketError.HostNotFound))

        Catch ex As Exception
            m_State = BazSocketState.Disconnected
            OnStateChanged(BazSocketState.Disconnected)
            OnConnectionFailed(ex)
        End Try


    End Sub

    Private Sub Connect_CallBack(ByVal Ar As IAsyncResult)
        Dim AsyncSocket As Socket = CType(Ar.AsyncState, Socket)

        'Regarde si le socket Existe encore, Sinon on quite
        If Not Object.ReferenceEquals(AsyncSocket, m_Socket) Then
            Return
        End If

        Try
            m_Socket.EndConnect(Ar)
            If Not m_Socket.Connected Then
                Throw New Exception()
            End If

            'Normalement ici on est connecter.
            m_Stream = New NetworkStream(m_Socket) 'Crée notre stream
            m_State = BazSocketState.Connected
            OnStateChanged(BazSocketState.Connected)
            OnConnected()

            m_Stream.BeginRead(m_RecvBuffer, 0, m_RecvSize, New
AsyncCallback(AddressOf Receive_CallBack), m_Socket)
        Catch ex As Exception
            m_State = BazSocketState.Disconnected
            OnStateChanged(BazSocketState.Disconnected)
```

```
            OnConnectionFailed(ex)
            Return
        End Try
    End Sub

    Private Sub Accept_CallBack(ByVal Ar As IAsyncResult)
        Dim AsyncSocket As Socket = CType(Ar.AsyncState, Socket)
        Dim NewSocket As Socket, NewStream As NetworkStream

        SyncLock AcceptSync
            'Regarde si le socket Existe encore, Sinon on quite
            If Not Object.ReferenceEquals(AsyncSocket, m_Socket) Then
                Return
            End If


            Try
                NewSocket = m_Socket.EndAccept(Ar)
                If Not NewSocket.Connected Then
                    NewSocket.Close()
                    NewSocket = Nothing
                    Throw New Exception()
                End If

                NewStream = New NetworkStream(NewSocket) 'Crée notre
stream

                OnAccepted(New AcceptRequest(NewSocket, NewStream))

                m_Socket.BeginAccept(New AsyncCallback(AddressOf
Accept_CallBack), m_Socket)
            Catch ex As Exception
                m_State = BazSocketState.Disconnected
                OnStateChanged(BazSocketState.Disconnected)
                OnAcceptFailed(ex)
                Return
            End Try
        End SyncLock
    End Sub

    Private Sub Receive_CallBack(ByVal Ar As IAsyncResult)
        Dim AsyncSocket As Socket = CType(Ar.AsyncState, Socket)
        Dim RecvSize As Integer

        SyncLock RecvSync

            'Regarde si le socket Existe encore, Sinon on quite
            If Not Object.ReferenceEquals(AsyncSocket, m_Socket) Then
                Return
            End If

            Try
                RecvSize = m_Stream.EndRead(Ar)
                If RecvSize <> 0 Then
                    Dim ByteData(RecvSize - 1) As Byte
                    Array.Copy(m_RecvBuffer, ByteData, RecvSize)

                    OnDataArrival(ByteData)


                    If m_Stream.CanRead Then
```

```
                        m_Stream.BeginRead(m_RecvBuffer, 0,
m_RecvSize, New AsyncCallback(AddressOf Receive_CallBack), m_Socket)
                    Else
                        DisconnectSocket(True)
                    End If

                Else
                    DisconnectSocket(True)
                End If
            Catch ex As Exception
                DisconnectSocket(True)
            End Try
        End SyncLock
    End Sub

    Private Sub Send_CallBack(ByVal Ar As IAsyncResult)
        Dim AsyncSocket As Socket = CType(Ar.AsyncState, Socket)
        'Dim SendSize As Integer
        'Static CurrentSize As Integer

        SyncLock SendSync
            If Not Object.ReferenceEquals(AsyncSocket, m_Socket) Then
                Return
            End If

            Try
                m_Stream.EndWrite(Ar)
            Catch ex As Exception
                DisconnectSocket(True)
            End Try
        End SyncLock
    End Sub
#End Region

#Region " Fonctions privées "
    'Detruit le Socket et en crée un nouveau
    Private Sub DisconnectSocket(ByVal RaiseClose As Boolean)
        CloseSocket()
        CreateSocket()

        OnStateChanged(BazSocketState.Disconnected)
        If RaiseClose Then
            OnClosed()
        End If
    End Sub
    'Detruit le Socket completement (Il est ensuite inutilisable)
    Private Sub CloseSocket()
        If m_Socket Is Nothing Then Return

        If m_Socket.Connected Then
            m_Socket.Shutdown(SocketShutdown.Both)
            If m_Stream IsNot Nothing Then m_Stream.Close()
        End If
        If m_Init Then
            m_Socket.Close()
            m_Socket = Nothing
            'On a fermer le socket il n'est donc plus initialisé !
            m_Init = False
        End If
        m_State = BazSocketState.Disconnected
    End Sub
```

```
    'Crée un nouveau socket
    Private Sub CreateSocket()
        m_Socket = New Socket(AddressFamily.InterNetwork,
SocketType.Stream, ProtocolType.Tcp)
        m_State = BazSocketState.Disconnected

        If m_Socket Is Nothing Then
            m_Init = False
        Else
            m_Init = True
        End If
    End Sub
    'Initialise les buffers d'envoie et de reception
    Private Sub InitBuffers()
        If m_Socket IsNot Nothing Then
            m_RecvSize = m_Socket.ReceiveBufferSize
            ReDim m_RecvBuffer(m_RecvSize - 1)

            m_SendSize = m_Socket.SendBufferSize
            ReDim m_SendBuffer(m_RecvSize - 1)
        End If
    End Sub

    Private Function DefaultLocalEP(ByVal LocalPort As Integer) As
IPEndPoint
        Dim HostEntry As IPHostEntry =
Dns.GetHostEntry(Dns.GetHostName())
        For Each IP As IPAddress In HostEntry.AddressList
            If IP.AddressFamily = AddressFamily.InterNetwork Then
                Return New IPEndPoint(IP, LocalPort)
            End If
        Next

        Return Nothing
    End Function

    Private Function DefaultLocalEP(ByVal LocalIp As String, ByVal
Port As Integer) As IPEndPoint
        Dim HostEntry As IPHostEntry = Dns.GetHostEntry(LocalIp)
        For Each IP As IPAddress In HostEntry.AddressList
            If IP.AddressFamily = AddressFamily.InterNetwork Then
                Return New IPEndPoint(IP, Port)
            End If
        Next

        Return Nothing
    End Function
#End Region

End Class
```

BazSocketState.vb : حالة الأداة سوكت Socket وهو عبارة عن Class.vb (كلاس) وتم استدعائه كما وضحنا في الصور السابقة ...

```
Public Enum BazSocketState

    Disconnected
    Resolving
    Listening
    Connecting
    Connected

End Enum
```

الشرح

حالة الإتصال . ← Disconnected

حالة الإستقبال . ← Resolving

حالة الاستماع . ← Listening

حالة الإتصال الآن . ← Connecting

حالة الإتصال الماضي . ← Connected

EventHandler.vb : الحالات ، وهو عبارة عن **Class.vb** (كلاس) وتم استدعائه كما وضحنا في الصور السابقة ...

```
Public Delegate Sub DataArrivalEventHandler(ByVal sender As Object,
ByVal data() As Byte)

Public Delegate Sub AcceptedEventHandler(ByVal sender As Object,
ByVal Request As AcceptRequest)

Public Delegate Sub ErrorEventHandler(ByVal sender As Object, ByVal
ex As Exception)

Public Delegate Sub SendProgressEventHandler(ByVal sender As Object,
ByVal current As Integer, ByVal total As Integer)

Public Delegate Sub SendCompleteEventHandler(ByVal sender As Object,
ByVal total As Integer)

Public Delegate Sub StateChangedEventHandler(ByVal sender As Object,
ByVal state As BazSocketState)
```

الشرح

حالة وصول البيانات .

```
Public Delegate Sub DataArrivalEventHandler(ByVal sender As Object,
ByVal data() As Byte)
```

حالة الاستقبال .

```
Public Delegate Sub AcceptedEventHandler(ByVal sender As Object,
ByVal Request As AcceptRequest)
```

حالة حدوث الأخطاء .

```
Public Delegate Sub ErrorEventHandler(ByVal sender As Object, ByVal
ex As Exception)
```

معالج الإرسال .

```
Public Delegate Sub SendProgressEventHandler(ByVal sender As Object,
ByVal current As Integer, ByVal total As Integer)
```

إكمال الإتصال .

```
Public Delegate Sub SendCompleteEventHandler(ByVal sender As Object,
ByVal total As Integer)
```

اختيار حالة السوكت Socket في اتصال أو استماع .

```
Public Delegate Sub StateChangedEventHandler(ByVal sender As Object,
ByVal state As BazSocketState)
```

Module1.vb : عبارة عن كود للقيام بعملية بدء الاتصال وقطع الاتصال و إرسال واستقبال

البيانات بين المسئول والعميل إلى مالا نهاية وقد قمنا باستدعائها كما وضحنا سابقاً في الصور ..

```
Imports System.Net.Sockets
Imports System.Text
Module Module1
    Dim clientsList As New Hashtable
    Sub Main()
        Dim serverSocket As New TcpListener(8888)
        Dim clientSocket As TcpClient
        Dim infiniteCounter As Integer
        Dim counter As Integer

        serverSocket.Start()
        msg("Chat Server Started ....")
        counter = 0
        infiniteCounter = 0
        For infiniteCounter = 1 To 2
            infiniteCounter = 1
            counter += 1
            clientSocket = serverSocket.AcceptTcpClient()

            Dim bytesFrom(10024) As Byte
            Dim dataFromClient As String

            Dim networkStream As NetworkStream = _
            clientSocket.GetStream()
            networkStream.Read(bytesFrom, 0,
CInt(clientSocket.ReceiveBufferSize))
            dataFromClient =
System.Text.Encoding.ASCII.GetString(bytesFrom)
            dataFromClient = _
            dataFromClient.Substring(0, dataFromClient.IndexOf("$"))

            clientsList(dataFromClient) = clientSocket

broadcast(dataFromClient + " Joined ", dataFromClient, False)

            msg(dataFromClient + " Joined chat room ")
            Dim client As New handleClinet
 client.startClient(clientSocket, dataFromClient, clientsList)
        Next

        clientSocket.Close()
        serverSocket.Stop()
        msg("exit")
        Console.ReadLine()
    End Sub

    Sub msg(ByVal mesg As String)
        mesg.Trim()
        Console.WriteLine(" >> " + mesg)
    End Sub
    Private Sub broadcast(ByVal msg As String, _
    ByVal uName As String, ByVal flag As Boolean)
```

```
            Dim Item As DictionaryEntry
            For Each Item In clientsList
                Dim broadcastSocket As TcpClient
                broadcastSocket = CType(Item.Value, TcpClient)
                Dim broadcastStream As NetworkStream = _
                        broadcastSocket.GetStream()
                Dim broadcastBytes As [Byte]()

                If flag = True Then
       broadcastBytes = Encoding.ASCII.GetBytes(uName + " says : " + msg)
                Else
                    broadcastBytes = Encoding.ASCII.GetBytes(msg)
                End If

                broadcastStream.Write(broadcastBytes, 0,
    broadcastBytes.Length)
                broadcastStream.Flush()
            Next
        End Sub

        Public Class handleClinet
            Dim clientSocket As TcpClient
            Dim clNo As String
            Dim clientsList As Hashtable

            Public Sub startClient(ByVal inClientSocket As TcpClient, _
            ByVal clineNo As String, ByVal cList As Hashtable)
                Me.clientSocket = inClientSocket
                Me.clNo = clineNo
                Me.clientsList = cList
    Dim ctThread As Threading.Thread = New Threading.Thread(AddressOf
    doChat)
            ctThread.Start()
            End Sub

            Private Sub doChat()
                Dim infiniteCounter As Integer
                Dim requestCount As Integer
                Dim bytesFrom(10024) As Byte
                Dim dataFromClient As String
                Dim sendBytes As [Byte]()
                Dim serverResponse As String
                Dim rCount As String
                requestCount = 0
                For infiniteCounter = 1 To 2
                infiniteCounter = 1
```

```
Try
  requestCount = requestCount + 1
   Dim networkStream As NetworkStream = _
clientSocket.GetStream()
networkStream.Read(bytesFrom, 0,
CInt(clientSocket.ReceiveBufferSize))
dataFromClient = System.Text.Encoding.ASCII.GetString(bytesFrom)
dataFromClient = _
dataFromClient.Substring(0, dataFromClient.IndexOf("$"))
msg("From client - " + clNo + " : " + dataFromClient)
rCount = Convert.ToString(requestCount)
  broadcast(dataFromClient, clNo, True)
   Catch ex As Exception
    MsgBox(ex.ToString)
                End Try
            Next
        End Sub

    End Class
End Module
```




شرح الأكواد الخاصة بواجهة العميل Client

يستخدم هذا الكود للشريط الذي يوجد أسفل الفورم الخاص بالعميل ليبين المراحل التي يمر فيها إثناء القيام بعملية الاتصال باللآيبي ، الشريط الذي يسمى StatusStrip وهو عبارة عن أداه موجودة ضمن أدوات Toolbox ..

```
Private Sub MySock_StateChanged(ByVal sender As Object, ByVal state
As BazSocketState) Handles MySock.StateChanged

Select Case state
     Case BazSocketState.Connected
      lbStatusConn.Text = "متصل حالياً"
        Button2.Enabled = False
        Case BazSocketState.Connecting
            lbStatusConn.Text = " يتم الاتصال حالياً"
        Case BazSocketState.Resolving
            lbStatusConn.Text = " جلب المعلومات "
        Case BazSocketState.Disconnected
            lbStatusConn.Text = " غير متصل "
             Button2.Enabled = True
            Case BazSocketState.Listening
                lbStatusConn.Text = " في حالة استماع "

        End Select


    End Sub
```

يستخدم هذا الكود عندما يقوم العميل بالإتصال بمنفذ الروتر عند الضغط على بوتون الـ Connection (الإتصال) .

```
Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button2.Click

    MySock.Connect(txtIP.Text, 1007)


    End Sub
```

تم بحمد الله تعالى